Complementary Incremental Hashing with Query-adaptive Re-ranking for Image Retrieval

Complementary Incremental Hashing with Query-adaptive Re-ranking for Image Retrieval

Xing Tian, Wing W. Y. Ng*, Hui Wang, Sam Kwong
*corresponding author

***Abstract*—Concept drift is prevalent in non-stationary data environments but is rarely researched in image retrieval. Therefore, more research is needed on image retrieval in non-stationary data environments so that highly relevant images can still be retrieved when concept drifts happen. Hashing is a key technique to allow efficient image retrieval, so incremental hashing technique emerges in recent years for image retrieval in non-stationary environments. A state-of-the-art method is *Incremental Hashing* (ICH). ICH trains new hash tables on new data without considering the performance of previous hash tables, so the dependency of successive hash tables is ignored. To make use of this dependency in order to improve the performance of image retrieval in non-stationary environments, *Complementary Incremental Hashing with query-adaptive Re-ranking* (CIHR) is proposed in this paper. CIHR trains multiple hash tables incrementally, one for each data chunk of images. A new hash table is trained on a new data chunk of images as well as those images badly hashed by previous hash tables, thus the new hash table is complementary to the previous hash tables. To use the hash tables more effectively, a query-adaptive re-ranking method is used to weight all hash functions in each hash table according to their retrieval performance with respect to a given query. Weighted Hamming distance is finally used to evaluate the similarity between the query and the images in the database, as the basis of image retrieval. Experimental results on simulated non-stationary scenarios show that the proposed CIHR method achieves higher retrieval accuracy than all methods being compared, thus setting a new state of the art in image retrieval in non-stationary data environments.**

***Index Terms*—Image Retrieval, Hashing, Non-stationary Environment, Concept Drift, Re-ranking.**

I. Introduction

IMAGE environments in the real world are mostly non-stationary as new images are constantly being added. For example, it has been reported that over 300 million images are uploaded to the Facebook website every day [1]. With new images being added, new classes may appear. Moreover, the data distribution of existing classes may also change. Then, the problem of *concept drift* [2], [3] arises, which is common in non-stationary data environments. Therefore, image retrieval techniques specifically designed to work in non-stationary data environments emerge, one of which is image hashing.

Image hashing is an advanced indexing technique which learns compact hash codes for images. However, most of the existing hashing methods focus on image retrieval in stationary environments which cannot adapt to changes in nonstationary data environments such as concept drift. Moreover, retraining hash model using whole dataset whenever new data arrives would be time consuming and impractical for real applications. Several hashing methods have been proposed recently for image retrieval in non-stationary data environments, such as Online Kernel-based Hashing (OKH) [4]. However, most of them only assume that data arrive in database consecutively without considering the changes in (class-specific) data distribution, i.e. concept drift. ICH is the only multi-table based image hashing method that deals with image retrieval in non-stationary data environment with concept drift, where hash tables are trained iteratively as data chunks arrive consecutively. However, in ICH, multiple hash tables are trained incrementally and independently. The similarity information between some pairs of images may not be preserved well by their hash codes, i.e., the similarity of images does not agree with their similarity by their hash codes in terms of their labels; thus these images are *badly hashed*. These images are not re-hashed in the subsequent training of new hash tables, thus the risk of image retrieval performance being lowered.

We hypothesis the badly hashed images should be re-hashed in subsequent training of new hash tables to improve image retrieval performance. Consequently, successive hash tables become dependent. To address this dependency issue, a new incremental hashing method, *Complementary Incremental Hashing with query-adaptive Re-ranking* (CIHR), is proposed in this paper. When a new chunk of images arrives, a new hash table (consisting of multiple hash functions) is trained based on the newest data chunk as well as those images badly hashed by previous hash tables. In this way, the new hash table is complementary to previous hash tables. It corrects errors caused by previous hash tables and preserves the similarity information in the newest data chunk in new hash functions.

For a given query, some hash functions may generate more discriminative hash codes than others. Therefore, a query-

Xing Tian (shawntian123@gmail.com) is with the Guangdong Provincial Key Laboratory of Computational Intelligence and Cyberspace Information, School of Computer Science and Engineering, South China University of Technology, Guangzhou, Guangdong, 510006, China; and also with the Department of Computer Science, City University of Hong Kong, Hong Kong, China.

Wing W.Y. Ng (wingng@ieee.org, corresponding author) is with the Guangdong Provincial Key Laboratory of Computational Intelligence and Cyberspace Information, School of Computer Science and Engineering, South China University of Technology, Guangzhou, Guangdong, 510006, China.

Hui Wang (h.wang@ulster.ac.uk) is with School of Computing, Ulster University, Belfast, UK.

Sam Kwong (cssamk@cityu.edu.hk) is with the Department of Computer Science, City University of Hong Kong, Hong Kong, China.

This work was supported in part by the National Natural Science Foundation of China under Grant 61876066, 61772344 and 61672443, Guangdong Province Science and Technology Plan Project (Collaborative Innovation and Platform Environment Construction) 2019A050510006, Hong Kong RGC General Research Funds under Grant 9042489 (CityU 11206317), Grant 9042816 (CityU 11209819) and Grant 9042322 (CityU 11200116), and EU Horizon 2020 Programme (700381, ASGARD).

TABLE I: Commonly used symbols List

i, j, r, b, k	Index variables	w_b^k	Projection vector of the b^{th} hash function in the k^{th} hash table
x	An image feature vector	W^T	One set of hash functions at time $t = T$
t	Serial number of time unit	$h_b(x_i)$	Hash value of the b^{th} hash function for image x_i
D^T	Data chunk appears at time $t = T$	$I_b(x_i)$	Hash bit value of the b^{th} hash function for image x_i
d	Dimensionality of an image feature vector	$H(x_i, x_j)$	Accumulated Hamming distance between images x_i and x_j
N_D	Number of images in one data chunk	A	Evaluated similarity matrix
D_l^T	Labeled image set in D^T	S	True similarity matrix
N_l	Number of labeled images in one data chunk	$E(x_i)$	Errors related to image x_i
G^T	Badly hashed image set selected at time $t = T$	φ^+	Similar image pair set
n	Number of badly hashed images selected	φ^-	Dissimilar image pair set
X_l^T	Labeled images in training set at time $t = T$	γ_i	Similarity consistency of the i^{th} hash table
X^T	Training set at time $t = T$	$f_i(x_j, x_r)$	Hamming distance between images x_j and x_r using the i^{th} hash table
l	Number of labeled images in training set	δ_i	Hash code variance of the i^{th} hash table
K	Number of hash tables employed	σ_b	Variance of the b^{th} hash bit
B	Number of hash functions in one hash table	v_i	Weight of the i^{th} hash table
Ω^T	The multi-hashing system generated at time $t = T$	$d(x_q, w_b)$	Euclidean distance between image x_q to the b^{th} hash hyperplane
α_s	Threshold parameter for similar images	$\beta(x_q, w_b^k)$	Query-adaptive weight for image x_q to hash function w_b^k
α_d	Threshold parameter for dissimilar images	$H_F(x_q, x_i)$	Weighted Hamming distance between image x_q and x_i

adaptive re-ranking method is also proposed in this paper. A query-adaptive weight is calculated for each hash function according to the distance between the query and the hash hyperplane implied by a hash function. Final Hamming distances between a query and candidate images are weighted by combining weighs of hash tables and query-adaptive weighs of hash functions. Commonly used symbols in this paper are listed in Table I.

Contributions of this paper are summarized as follows:

- A novel incremental hashing method which trains complementary hash tables for non-stationary data environment is proposed. New complementary hash table is trained incrementally with badly hashed images by previous hash tables. Moreover, hash tables cannot adapt to current data environment will be removed.
- A query-adaptive re-ranking method is proposed in this paper. Query-adaptive weights for all hash functions in each hash table are computed according to the location of the query with respect to hash hyperplanes. Weighted Hamming distances between the query and candidate images are computed to evaluate their similarities.
- The proposed method and comparative methods are evaluated on simulated non-stationary scenarios. In all non-stationary scenarios, the proposed method achieves superior performance over other state-of-the-art non-stationary hashing methods consistently.

This paper is organized as follows. In Section II, related works are introduced. The proposed method is described in detail in Section III. Section IV shows and discusses experimental results of the proposed method and other comparative methods on simulated non-stationary scenarios. We conclude the paper in Section V.

II. Related Works

Hashing methods learn hash functions to project the original data into the low dimensional Hamming space with similarity preserved. According to whether semantic information is used for training, hashing methods can be generally categorized into three types: unsupervised, supervised, and semi-supervised hashing methods. Moreover, a few non-stationary hashing methods are also proposed in recent years to handle the image retrieval task in non-stationary environments, which is also the target problem issued in this paper. These methods will be also elaborated in this section.

A. Hashing Methods for Stationary Environments

The unsupervised hashing methods learn hash functions without using the label of images. The Locality Sensitive Hashing [5] and its variants, such as SKLSH [6], KLSH [7], and LSH for Chi2 distance [8], are representative unsupervised methods which generate hash functions randomly. Spectral Hashing [9], [10] generates compact hash codes for images by solving the problem of graph partitioning. Spherical Hashing [11] employs hash hyperspheres rather than hyperplanes as hash functions. The unsupervised visual hashing with semantic assistant [12] extracts semantic similarity information from the related documents of images to train hash functions. Unsupervised local feature Hashing [13] generates hash code of images by bilinear projection based on the local feature of images. By utilizing the sparse reconstructive relationship of data, hash functions are trained in the Special Structure-based Hashing [14]. Sparse Hashing with Optimized Anchor Embedding [15] selects optimal anchor points for the sparse representation of images in Hamming space. Distributed graph hashing is proposed in [16] which learns hash functions in a distributed manner. Anchor Graph Hashing (AGH) [17]

utilizes the anchor graph of data to directly approximate the adjacency matrix and build graph Laplacians. Manhattan hashing (MH) [18] encodes each hash projections with multiple hash bits for hash quantization to preserve the neighborhood structure information of data in original feature space. Both AGH and MH generate multiple hash bits for each hash projections. There are also some retrieval methods employing re-ranking and query-adaptive techniques to further improve their retrieval performance [19]–[24]. A web image search re-ranking approach is proposed in [19] to explore multiple modalities in a graph-based learning scheme. A cross-modality image search scheme [20] projects concept prototypes and image features to the latent semantic space to minimize the semantic gap. The query-adaptive hash code ranking method [21] generates query-adaptive weight and uses weighted Hamming distance for final similarity evaluation. The bit selection hashing method [22], [24] selects optimal hash bit combination from various hash bits to generate the compact hash code. However, these query-adaptive hashing methods generally need to find nearest samples to each query in advance for query-adaptive weighting, which are time consuming for real applications.

Supervised hashing methods utilize label information of images to improve the semantic retrieval performance. Spectral Hashing with semantically consistent graph [25] learns hash codes by optimizing the graph Laplacian directly, without using Euclidean distance to construct the graph Laplacian. Nonlinear Discrete Hashing [26] generates compact hash codes for images in database by employing a nonlinear neural network with multiple layers. Multimodal Discriminative Binary Embedding [27] trains different sets of hash functions for each type of data and generates discriminative hash codes. Supervised Matrix Factorization Hashing [28] trains hash projection matrix for each modal of data by considering both the semantic similarity relationship and local geometric structure information. Spectral Multimodal Hashing [29] only utilizes similar pair of data in different models for the training. Recent years have also witnessed the emergence of deep hashing methods which learns compact hash codes based on high-level features of images. Deep supervised hashing [30] learns hash codes based on the similarity information of image pairs. Another deep hashing method is proposed in [31] which learns instance-aware representations for multi-labeled images and generates multiple pieces of hash codes for each image. Triplet-based deep hashing [32] learns discriminative hash codes based on the relative similarity information between heterogeneous data for cross-modal retrieval.

Semi-supervised hashing methods trains hash functions based on partially labeled database. The Sequential Projection Learning Hashing [33] trains hash functions sequentially. Each hash function is learned by increasing the weights of wrongly predicted image pairs by its previous hash function. In Bootstrap Sequential Projection Learning Hashing (BSPLH) [34], each hash function is trained by correcting the error caused by all previous hash functions holistically. Based on Complementary Hashing [35], the Dual Complementary Hashing (DCH) method [36] trains complementary hash functions and hash tables simultaneously. A drawback of the DCH is that correctly predicted image pairs will not be used for the following training of next hash tables. To address this issue, BBSHR [37] is proposed which keeps correctly predicted image pairs and increases the weights of badly predicted image pairs for training.

B. Existing Non-stationary Hashing Methods

Most of current hashing methods retrieve relevant images in a stationary database, while the data environment in real world is usually non-stationary. New appearing images may cause changes of the data distribution of existing classes. Images of new classes may also appear. In the non-stationary data environment, the performance of stationary hashing methods would degrade, since they cannot update hash table according to the new appearing data. In recent years, image retrieval problem in non-stationary data environment has attracted more and more attentions. Several non-stationary hashing methods have been proposed.

The OKH method [4] updates hash functions based on pairs of images appearing over time with their similarity information. The Online Supervised Hashing [38] updates hash functions with newly appearing labeled data by using Error Correcting output codes. In [39], adaptive Hashing updates hash functions based on stochastic gradient descent with pairs of images appearing over time. MIHash [40] employs the mutual information between similarity and Hamming distance distributions as the objective function for the training of hash functions. The OSH [41] method learns hash functions based on the sketch of database, which could be updated dynamically with new data appearing over time. Based on the OSH method, OSSH is proposed in [42] which learns hash functions based on a semantic graph. Based on the OKH method, the OH method [43] is proposed which updates hash functions by minimizing the prediction loss function. The IBL method [44] trains new hash functions over time and selects optimal set of hash functions according to the current data environment. The Incremental Hashing (ICH) [3] employs a multi-hashing system which contains multiple hash tables and corresponding weights. Whenever new chunk of data appears, the multi-hashing system in ICH will be updated.

Incremental Hashing is the only existing semi-supervised non-stationary hashing method which employs multiple hash tables for image retrieval with concept drift happening. However, hash tables in the multi-hashing system of ICH are trained independently. The badly hashed images by the previous hash tables will be ignored for the training of new hash table, but they are still used as candidate images for the following retrieval. Therefore, in this paper, Complementary Incremental Hashing method with query-adaptive Re-ranking (CIHR) is proposed which trains complementary hash tables for retrieval in the non-stationary data environment. The badly hashed images by previous hash tables will be used to form the training set of new hash table. Moreover, in ICH, the hash tables are weighted only, while different hash functions actually contribute differently to the similarity evaluation for a given query. In this paper, a query-adaptive re-ranking method is also introduced to weight all hash functions in the multi-hashing system to further improve retrieval accuracy.

III. Complementary Incremental Hashing with Query-adaptive Re-ranking

The complementary incremental hashing method with query-adaptive re-ranking (CIHR) consists of three parts: badly hashed images selection, multi-hashing system training and weighting, and query-adaptive re-ranking, which will be introduced in the following subsections sequentially. The overview of CIHR at time $t = T$ is shown in Figure 1, which consists of both training procedures and retrieval procedures. Before a new data chunk appears at time $t = T$, a set of badly hashed image has already been selected based on the existing multi-hashing system in CIHR. This badly hashed image set is combined with the new data chunk to build the training set for a new hash table. The multi-hashing system in CIHR consisting of both hash tables and corresponding weights is then updated. Finally, based on the current updated multi-hashing system in CIHR, new badly hashed image set is selected for training in the next time unit. In retrieval procedures, for a given query, the query-adaptive weight for each hash function is computed according to the location of the query with respect to each hash hyperplane. Weighted Hamming distances between this query and all candidate images in the database are then computed. Images yielding the smallest Hamming distances are returned as the final retrieval results.

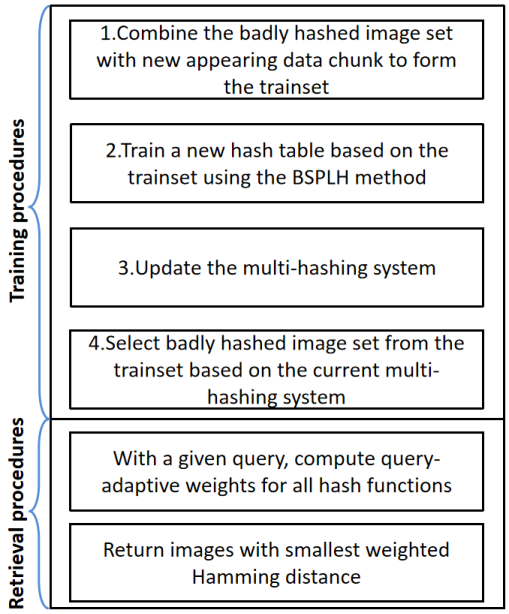


Fig. 1: The overview of CIHR at time t=T.

In the rest of this section, we firstly introduce how to select the badly hashed images by previous hash tables in subsection III-A. Then, the training and weighting methods of individual hash table in CIHR are introduced in subsection III-B. Subsection III-C describes the query-adaptive re-ranking method proposed in this paper.

A. Selection Method for Badly Hashed Images by Previous Hash Tables

Without loss of generality, assume that there is a new data chunk appearing at each time unit, with same number of images totally and same number of labeled images. The data chunk appearing at time $t = T$ is denoted as $D^T \in R^{d\times N_D}$, where d and N_D denote the dimensionality of the image feature vector and number of images in this data chunk, respectively. Let $D_l^T \in R^{d\times N_l}$ denotes the labeled image set in D^T, where N_l denotes the number of labeled images. At time $t = T$, the badly hashed image set $G^{T-1} \in R^{d\times n}$ is selected at the beginning and combined with the newest data chunk D^T to form the training set X^T, where n denotes the number of badly hashed images selected at each time unit. The training set X^T is used for the training of the newest hash table. Since all images in G^{T-1} are labeled, the labeled image set X_l^T in X^T contains both the badly hashed image set G^{T-1} and labeled image set D_l^T, i.e. $X_l^T = D_l^T \bigcup G^{T-1}$, where $X_l^T \in R^{d\times l}$, $l = N_l + n$. The multi-hashing system of CIHR generated at time T is denoted as Ω^T.

Assume at time $t = T - 1$, a new hash table is trained based on the training set X^{T-1} with labeled image subset X_l^{T-1}. This new hash table is finally added into the multi-hashing system, which contains K hash tables in total. Each hash table contains B hash functions. The hash value of the b^{th} hash function for an image x_i can be computed as follows:

$$h_b(x_i) = sgn(w_b' x_i) \tag{1}$$

where $sgn(\cdot)$ and w_b denote the sign function and projection vector of the b^{th} hash function, respectively. w_b' denotes the transpose of the projection vector. Since the value of sign function belongs to $\{-1,1\}$, to transfer hash value into binary hash bit, the hash bit value of the b^{th} hash function for an image x_i can be computed as follows:

$$I_b(x_i) = \frac{1}{2}(1 + h_b(x_i)) \tag{2}$$

Two similar images are expected to have small Hamming distance, while two dissimilar images should have large Hamming distance. If the similarity relationships between one image and many of other images are evaluated wrongly based on their Hamming distances, this image are regarded as badly hashed by these hash tables. To judge whether the similarity relationship between two images are evaluated correctly, the label information is required. Therefore, the labeled image subset, i.e. X_l^{T-1}, in the training set X^{T-1}, is utilized for selecting badly hashed images based on previous hash tables. The Hamming distances between any two images in X_l^{T-1} are computed, based on the multiple hash tables in Ω^{T-1}. To evaluate the similarity relationship between two images based on the existing K hash tables, the accumulated Hamming distance between two images x_i and x_j in X_l^{T-1} based on all hash tables is computed as follows:

$$H(x_i, x_j) = \sum_{k=1}^{K}\sum_{b=1}^{B}\{I_b(x_i) \oplus I_b(x_j)\} \tag{3}$$

where $H(\cdot,\cdot)$ and $\oplus$ denote the Hamming distance function between two images and the XOR operation between two binary hash bits, respectively. In Eq. (3), the Hamming distance is computed without using the weights of hash tables. The reason is that weights of hash tables are generated according to their performance to the current data environment, while the Hamming distance in Eq. (3) is used to evaluate whether the images are partitioned correctly by the original hash hyperplanes. Thus, accumulated Hamming distance between any two labeled images in X_l^{T-1} is computed without using weights of hash tables.

According to the Hamming distance computed between any two images in X_l^{T-1}, the similarity relationship of any two images can be evaluated. Two images are judged as similar, if their Hamming distance is smaller than a threshold Hamming distance. If the Hamming distance between two images is larger than another threshold Hamming distance, they are judged as dissimilar. We set two parameters α_s and α_d for two threshold Hamming distances, where $0 < \alpha_s < \alpha_d < 1$. For two images with Hamming distance between two threshold Hamming distances, it is difficult to judge their similarity relationship. Thus, the similarity relationship between these images are not evaluated in the proposed method. Let the matrix $A \in R^{l\times l}$ records the evaluated similarity relationships between any two images in X_l^{T-1}. The elements in A can be computed as follows:

$$A_{ij} = \begin{cases} 1, & H(x_i,x_j) \le \alpha_s \cdot KB \\ -1, & H(x_i,x_j) \ge \alpha_d \cdot KB \\ 0, & \alpha_s \cdot KB \le H(x_i,x_j) \le \alpha_d \cdot KB \end{cases} \tag{4}$$

Moreover, according to the labels of images in X_l^{T-1}, the true similarity relationship between any two images can be obtained. If two images share the same label, they are regarded as similar. Otherwise, they are dissimilar. In this way, the true similarity matrix S can be generated with elements defined as follows:

$$S_{ij} = \begin{cases} 1, & x_i \; and \; x_j \; are \; similar \\ -1, & x_i \; and \; x_j \; are \; dissimilar \end{cases} \tag{5}$$

By comparing the evaluated similarity relationship matrix A with the true similarity relationship matrix S, the badly hashed labeled images can be selected. For one image x_i, the i^{th} row vector in matrix A, denoted as A_i, records the evaluated pairwise similarity relationships between this image and all other labeled images in X_l^{T-1}. Similar, the i^{th} row vector in S, denoted as S_i, records the true similarity relationships. As aforementioned, if the similarity relationships between x_i and many of other images are evaluated wrongly based on their Hamming distances comparing to the true similarity relationships, this image is regarded as badly hashed. Thus, the errors related to the image x_i caused by the existing multi-hashing system Ω^{T-1} can be computed as follows:

$$E(x_i) = ||A_i - S_i||_1 \tag{6}$$

where $E(\cdot)$ and $||\cdot||_1$ denote the error function for one image, and the ℓ_1 norm function, respectively. Finally, n images with largest error value in X_l^{T-1} are selected to form the badly hashed image set G^{T-1}. This labeled database will be utilized at time $t = T$ to be combined with the newest data chunk to form the labeled training set X_l^T. Specially, at time $t = 0$, since there is no old data existing, badly hashed image set is regarded as empty set.

B. Training and Weighting of Hash Tables in CIHR

At time $t = T$, with the already selected badly hashed image set G^{T-1}, the training set X^T is generated and used to train a new hash table by BSPLH [34]. For simplify, the superscript time T is ignored in rest of this subsection. The hash functions are trained by maximizing the objective function as follows:

$$\max_W tr(W'X_lSX_lW) + \lambda tr(W'XX'W) \tag{7}$$

where W, $tr(\cdot)$, and λ denote the hash functions in the hash table, trace function, and the parameter of the regularization term, respectively. In Eq. 7, the first term denotes the empirical accuracy according to the semantic information, which maximizes the information contained in each hash bit. The second term is a regularizer used to control overfitting.

In CIHR, a multi-hashing system is employed, which contains K hash tables and their corresponding weights. After the new hash table being trained, the weights of all $K + 1$ hash tables are computed, which is similar to the idea in [3]. Each hash table is weighted according to their performance to the newest data chunk which represents the current data environment. Two aspects of performance of each hash table are taken into consideration, i.e. the Similarity Consistency (SC) and Hash Code Variance (HCV). For an ideal hash table, images sharing the same label are supposed to have small Hamming distance. Images belonging to different classes are dissimilar and supposed to have large Hamming distance. Let $f_i(x_j, x_r)$ denote the Hamming distance between two images x_j and x_r using the i^{th} hash table in the multi-hashing system. Then the Similarity Consistency of the i^{th} hash table is computed as follows:

$$\gamma_i = \sum_{(x_j,x_r)\in\varphi^+} f_i(x_j,x_r) - \sum_{(x_j,x_r)\in\varphi^-} f_i(x_j,x_r) \tag{8}$$

where φ^+ and φ^- denote the similar and dissimilar image pair set, respectively. Moreover, each hash table contains B hash functions. Each hash function can be regarded as a hash hyperplane that partitions the original data feature space into two parts. According to information theory, the entropy of one hash bit is maximized when the hyperplane partitions the original database in balance. In this case, the variance of this hash bit is maximized as well. For the i^{th} hash table, its Hash Code Variance is the summation of the variance of each hash bit, as follows:

$$\delta_i = \sum_{b=1}^{B} \sigma_b \tag{9}$$

where σ_b denotes the variance of the b^{th} hash bit. Both of the above two weights are normalized to [0, 1]. Then, the final

weight v_i of the i^{th} hash table is the product of these two aspects of weights, represented as follows:

$$v_i = \gamma_i \delta_i \tag{10}$$

Details of training procedures are shown in Figure 2. The blocks in this figure with blue or green color represent state of the system at time $t = T-1$ or T, respectively. The serial numbers of 1 to 4 in Figure 2 correspond to the steps in training procedures of CIHR shown in Figure 1. Firstly, with the set of previously selected badly hashed images G^{T-1} and the new appearing data chunk D^T, a training set X^T is generated at time $t = T$ by combining G^{T-1} and D^T. Secondly, a new set of hash functions W_0^T is learned with its weight v_0^T computed using X^T. Thirdly, the existing multi-hashing system Ω^{T-1} is updated to be Ω^T by removing the hash table yielding the least weight and adding the new learned hash table. Finally, the badly hashed image set G^T is selected based on the Pairwise similarity matrix S and the evaluated similarity matrix A for the training in the next time unit.

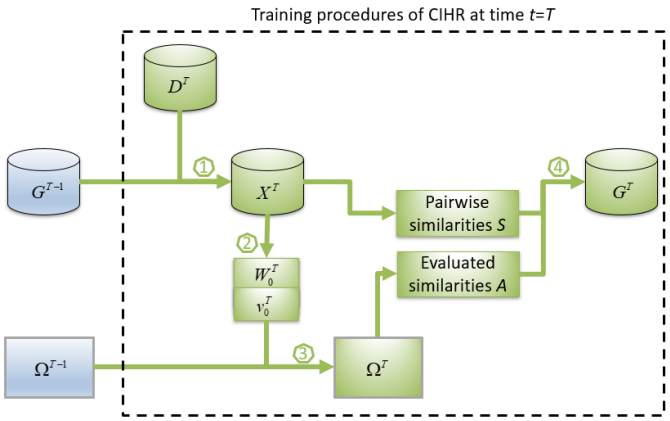


Fig. 2: Training procedures of CIHR at time t=T.

C. Query-Adaptive Re-ranking Method

In the CIHR method, hash tables in the multi-hashing system are firstly weighted according to their performance to the current data environment, as aforementioned. Moreover, for a given query, the performances of different hash functions in multiple hash tables are also different.

Note that Hamming distance is finally utilized to evaluate the similarity, which counts the number of different bits between two hash codes. In other words, the difference between two hash codes of images is computed to evaluate the difference between two images. For one hash function, if two images located on the different sides of the hyperplane, their corresponding hash bits are different, and contribute 1 Hamming distance to the final Hamming distance. For hash hyperplanes which are very close to the query, the similar (neighbor) images of the query could be partitioned in another side of this hash hyperplane. But for the hash hyperplanes which are far away to the query, if an image is located in the different side of the hyperplane to the query, this image must be also far from the query, which is dissimilar to the query. Based on this knowledge, the difference of hash bits caused by the hyperplanes, which are far to the query, is more convincing to compute the final Hamming distance as the difference between images.

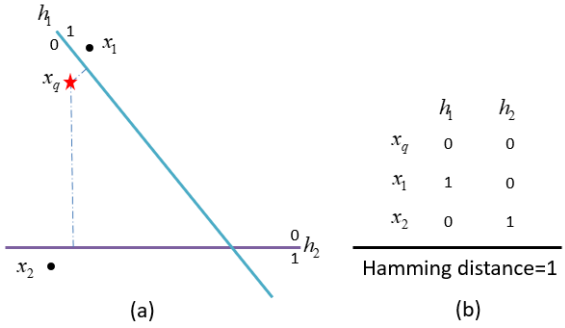


Fig. 3: Similar and dissimilar image to the query with same Hamming distance.

As shown in Figure 3(a), the query image x_q is close to the image x_1, but farther to the image x_2. There are two hash hyperplanes h_1 and h_2, where h_1 is very close to the query image x_q but h_2 is not. Since the hash hyperplane h_1 is very close the query image x_q, the neighbor x_1 is partitioned in the different sides of the query x_q. This shows that the Hamming distance caused by this bit may be not strong enough to reflect the difference between two images. However, for the hash hyperplane h_2, if one image is located at the different side of the query image x_q, then the difference of this hash bit is more convincing to denote the difference between two images. In Figure 3(b), the Hamming distance between x_q and x_1 is 1, due to the difference of the first bit, which is generated by the hash function h_1. The Hamming distance between x_q and x_2 is also 1, due to the second bit, which is generated by the hash function h_2. In fact, x_1 is very relevant to x_q, while x_2 is not. This figure shows that the Hamming distance caused by the hash hyperplane which is farther to the query is more convincing and more valuable, comparing to the hash hyperplanes which are close to the query image. Therefore, for a given query x_q, the Euclidean distance between one query and each hash hyperplane is computed as follows:

$$d(x_q, w_b) = \frac{x_q \cdot w_b}{||w_b||} = x_q \cdot w_b \tag{11}$$

where $d(\cdot,\cdot)$, $||\cdot||$, and $x_q \cdot w_b$ denotes the Euclidean distance function between image feature vector to one hash hyperplane, the norm of hash vector, and the dot product between the feature vector and norm vector of the hash hyperplane, respectively. Distances between the query to all hash hyperplanes in each hash table of the multi-hashing system are computed. Let d_{max} and d_{min} denote the maximum and minimum distance values. The query-adaptive weight for the query x_q to the b^{th} hash function in the k^{th} hash table, i.e. w_b^k, is normalized to [0,1] as follows:

$$\beta(x_q, w_b^k) = \frac{d(x_q, w_b^k) - d_{min}}{d_{max} - d_{min}} \tag{12}$$

where $\beta(x_q, \cdot)$ denotes the query-adaptive weight for a given query to one hash function. With weights for hash tables and query-adaptive weight for the query to each hash function, the final weighted Hamming distance between the query x_q to a candidate image in the database is computed as follows:

$$H_F(x_q, x_i) = \sum_{k=1}^{K} v_k \{ \sum_{b=1}^{B} \beta(x_q, w_b^k)(I_b^k(x_q) \oplus I_b^k(x_i)) \} \quad (13)$$

Finally, the candidate images in the database with smallest weighted Hamming distance to the query are returned as the final retrieval results.

At time $t = T$, the training in CIHR mainly consists of three steps: 1) training new hash functions; 2) updating weights of hash tables; 3) selecting badly hashed images. Time complexities of these three steps are $O(N_D B d^2 + B d l^2)$, $O(K N_D B + K B d l^2)$, and $O(B d l^2)$, respectively. Thus, the time complexity of training procedures in CIHR is $O(N_D B d^2 + K B d l^2)$. For a given query image, retrieval procedures for CIHR mainly consist of two steps: 1) computing query-adaptive weights; 2) computing weighted Hamming distances between query image and candidate images in database. Time complexities of these two steps are $O(KdB)$ and $O(KdBN_DT)$. The final time complexity of retrieval procedures in CIHR is $O(KdBN_DT)$.

IV. Experimental Results

The contents in this section are organized as follows. The basic settings in our experiments are introduced in Section IV-A. Section IV-B introduces the simulation methods for new class appearing scenarios, and shows the experimental results. The simulation methods for data distribution drifting scenarios and corresponding experimental results are shown in Section IV-C. Section IV-D shows the simulation methods of three more complicated combined scenarios and experimental results. In Section IV-E, experimental results on three non-stationary scenarios without concept drift are shown. Section IV-F shows the improvements of the CIHR comparing to the original ICH, and two simply modified versions of the stationary hashing methods. The values of parameters in CIHR are selected in Section IV-G.

A. Basic Settings of the Experiments

Four real world image databases, i.e. CIFAR-10, MNIST, CIFAR-100 and CACD, are utilized in our experiments to simulate non-stationary scenarios with concept drift problem. Two concept drift situations are considered in our experiments: new class appearing and data distribution drifting. There are 6 scenarios for new class appearing, 6 scenarios for data distribution drifting, and 3 scenarios for more complicated combined scenarios. These non-stationary data scenarios have been uploaded for public use[1]. The CIFAR-10 and MNIST databases are commonly used image databases and utilized to simulate the new class appearing scenarios. The CIFAR-100 and CACD are utilized to simulate scenarios for data distribution drifting. The CIFAR-10 database consists of $60,000$ real world color images, with 32×32 pixels each. These images belong to 10 classes evenly. Each image is described by a 512-dimension GIST feature vector. The MNIST database consists of $70,000$ grayscale handwritten digital images belonging to 10 classes, i.e. 0, 1, ..., 9. Each image has 28×28 pixels and is described by a 784-dimension feature vector (pixels). The CIFAR-100 database consists of $60,000$ real world color images belonging to 100 classes. Moreover, these 100 classes (a.k.a. subclass) can be further grouped into 20 superclasses evenly. Each image has 32×32 pixels and is represented as a 512-dimension GIST feature vector. The Cross-Age Celebrity Dataset (CACD) [45] consists of $163,446$ face images belonging to $2,000$ celebrities. Face images of each celebrity are drawn from the period of 10 years. In our experiments, we first regularize the size of each face images. Then the Multi-task CNN method (MTCNN) [46] is employed for face alignment. Each face image is finally represented by a 128-dimension feature vector, based on the FaceNet [47]. An open-source implementation of FaceNet[2] is utilized in our experiments (model 20170512-110547).

In our experiments, we compare the CIHR method with representative hashing methods, which contains stationary hashing methods, i.e. LSH, SH, SPLH, BSPLH, and non-stationary hashing methods, i.e. OKH, OSH, ICH, OH, IBL. The SPLH and BSPLH are representative stationary semi-supervised hashing methods. Moreover, the CIHR method also employs BSPLH method to train individual hash tables, which makes BSPLH very related to the work in this paper. The LSH and SH are representative unsupervised hashing methods. They are used as the baseline methods in our experiments. The OKH and OH are supervised online hashing methods which updates hash functions based on new appearing image pairs with their pairwise similarities. The OSH is an unsupervised online hashing method. The ICH and the IBL are the only two existing hashing methods that try to solve the image retrieval problem in non-stationary data environments with concept drift, which is also the problem we intend to solve in this paper. The parameters in all comparative methods are selected according to the original papers or the publicly released official codes. For the proposed CIHR method, we set $K = 5$, $B = 64$, $n = 100$, $\alpha_s = 0.1$, and $\alpha_d = 0.9$ in all experiments. For the supervised online hashing methods, i.e. OKH and OH, the labels of all images in the data chunk are utilized for training. Moreover, all experiments on non-stationary scenarios are repeated 10 times to get the average performance as the final experimental results.In this paper, we evaluate the retrieval performance of hashing methods according to their Top 100 precision and Top 1% precision [3], [44] based on the semantic groundtruth.

B. New Class Appearing Scenario

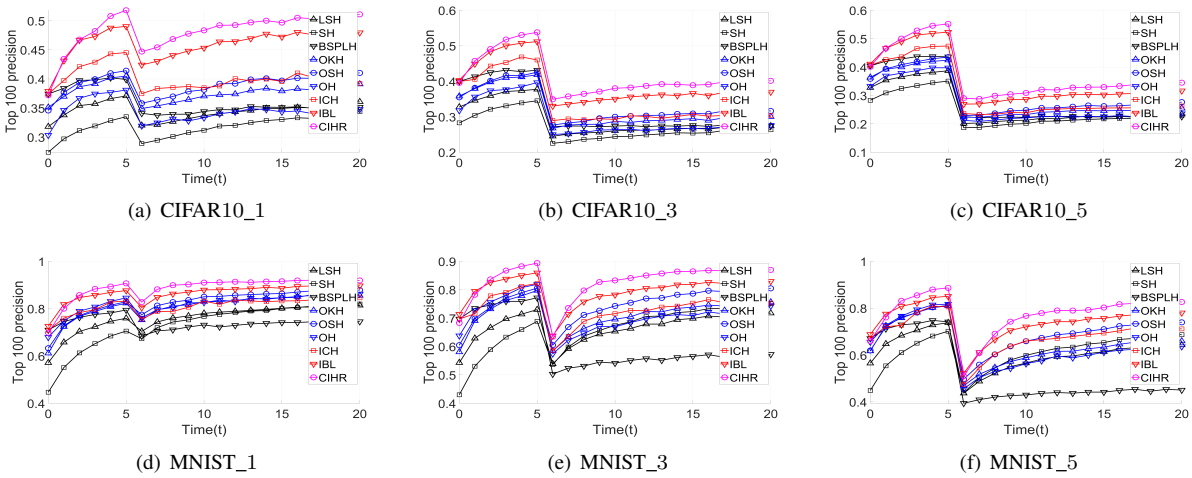
There are 6 non-stationary scenarios are simulated for new class appearing in our experiments, based on two real world image databases, i.e. CIFAR-10 and MNIST. Both of two databases contain 10 classes. In our experiments, there 21 chunks of data are generated for the period $t = 0 \sim 20$. At

[1] https://pan.baidu.com/s/15–vAlxe5Gbxwbd5roLFtA

[2] https://github.com/davidsandberg/facenet

TABLE II: Settings of 6 simulated scenarios for new class appearing.

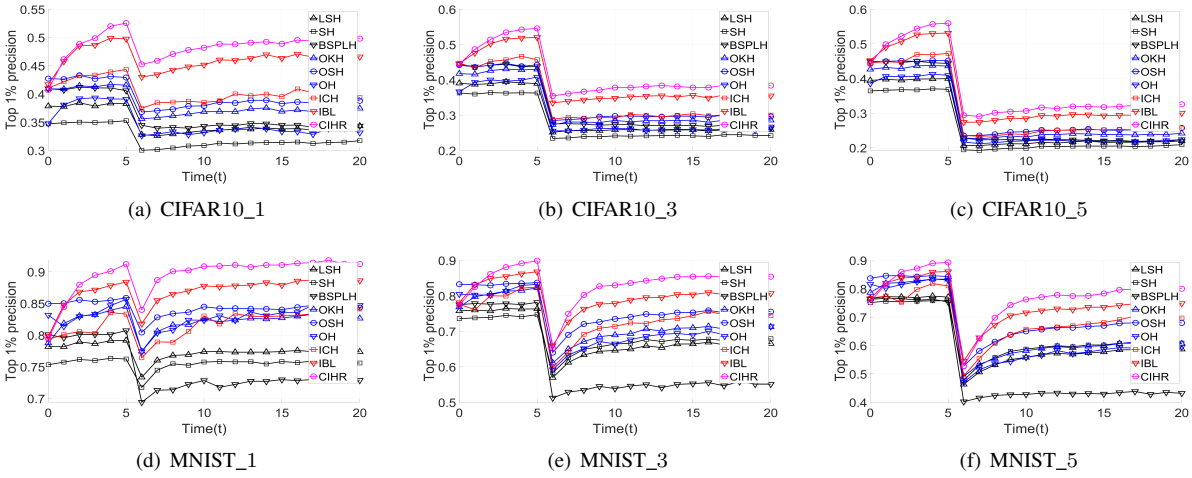
Database	$0 \leq t \leq 5$	$t \geq 6$	**Scenario name**
CIFAR-10	5 original appearing classes	5 original appearing classes and 1 newly appearing class	CIFAR10_1
	Same as above	5 original appearing classes and 3 newly appearing classes	CIFAR10_3
	Same as above	5 original appearing classes and 5 newly appearing classes	CIFAR10_5
MNIST	5 original appearing classes	5 original appearing classes and 1 newly appearing class	MNIST_1
	Same as above	5 original appearing classes and 3 newly appearing classes	MNIST_3
	Same as above	5 original appearing classes and 5 newly appearing classes	MNIST_5



(a) CIFAR10_1 (b) CIFAR10_3 (c) CIFAR10_5

(d) MNIST_1 (e) MNIST_3 (f) MNIST_5

Fig. 4: Top 100 precision on 6 new class appearing scenarios.



(a) CIFAR10_1 (b) CIFAR10_3 (c) CIFAR10_5

(d) MNIST_1 (e) MNIST_3 (f) MNIST_5

Fig. 5: Top 1% precision on 6 new class appearing scenarios.

each time unit, a new data chunk appears. Meanwhile, a series of test data chunk are also generated in the same manner. Each chunk of new appearing data and test set consists of $1,000$ images. For the new appearing data chunk, 100 images in it are labeled. To simulate the situation new class appearing, 5 of 10 classes are randomly selected as the original appearing classes since the time $t = 0$. Images in the chunk of new appearing data and test set are randomly selected in these 5 classes without replacement. Then, since time $t = 6$, images belonging to several (i.e. 1, 3, 5) new class(es) appear. Images used to generate data chunks are randomly selected in both the original appearing classes and new appearing class(es). The settings of 6 simulated scenarios for new class appearing are shown in Table II.

The experimental results of the CIHR method and other comparative methods are shown in Figure 4 and 5. According to Figure 4 and 5, there is significant performance drop at time $t = 6$, when the images of new classes appear. The stationary hashing methods, such as the LSH, SH and BSPLH, cannot adapt to the non-stationary data environment. Thus,

TABLE III: Settings of 6 simulated scenarios for data distribution drifting.

Database	$t = 0$	$t \geq 1$	Scenario name
CIFAR-100	20 superclasses	Subclasses changes according to Figure 7.	CIFAR100_20
	15 superclasses	Same as above	CIFAR100_15
	10 superclasses	Same as above	CIFAR100_10
	Same as above	Subclasses of 6 superclasses changes according to Figure 7.	CIFAR100_10_6
	Same as above	Subclasses of 3 superclasses changes according to Figure 7.	CIFAR100_10_3
CACD	All images with relative age equal 0	Data chunks with different relative ages appear according to Figure 9.	CACD

their performances are the worst comparing to other non-stationary hashing methods. The online hashing methods, i.e. OKH, OSH, and OH, could update hash functions according to the new appearing data. They achieve higher retrieval accuracy. However, the OKH and OH updates hash functions based on pairs of new appearing data. For a chunk of data with $1,000$ images, hash functions are updated 500 times. Only 500 pairwise similarities between images are utilized for training. The OSH is unsupervised and updates hash functions based on the data distributions only. Therefore, their performance is worse than the ICH and the IBL methods. The CIHR employs complementary multiple hash tables with query-adaptive re-ranking, which achieves the optimal retrieval performance.

Fig. 6: Drifting of the five subclasses for the superclass *Trees*.

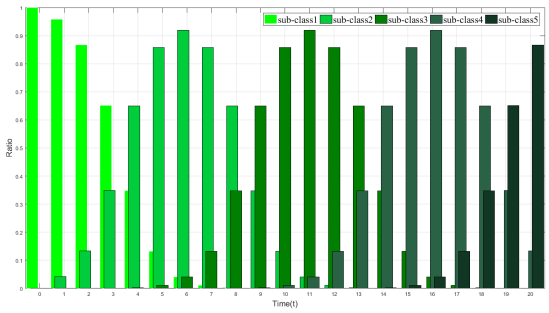


Fig. 7: Appearance ratio of five subclasses for one superclass in each data chunk over time.

C. Data Distribution Drifting Scenarios

There are 6 non-stationary scenarios are simulated for the situation data distribution drifting based on two image databases, i.e. CIFAR-100 and CACD. The CIFAR-100 database contains images belonging to 100 subclasses, which can be further grouped into 20 superclasses. For example, the superclass *Trees* consists of five subclasses, i.e. *maple*, *oak*, *palm*, *pine*, and *willow*, as shown in Figure 6. By controlling the appearance ratio of subclasses in data chunks over time as shown in Figure 7, which is similar to [3] [44], the data distribution drifting of superclasses can be simulated. 5 scenarios are simulated based on CIFAR-100. The first 3 scenarios employ different number (i.e. 20, 15, and 10) of superclasses since time $t = 0$, where the appearances ratio of subclasses are controlled over time as shown in Figure 7. The other two scenarios randomly select images from 10 superclasses, where the appearance ratio of subclasses in only several (i.e. 6 and 3) superclasses change over time. 21 data chunks are generated for the period $t = 0 \sim 20$, which contain $1,000$ images each. 100 images in each chunk are set labeled. Moreover, at each time unit, a test set containing $1,000$ images is also generated in the same manner of the new appearing data chunk.

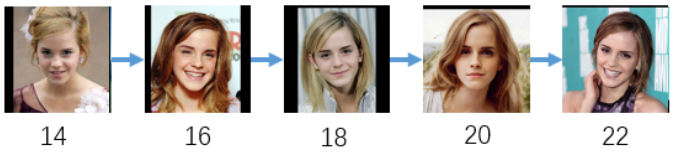


Fig. 8: Data distribution drifting in face images for one celebrity with different ages.

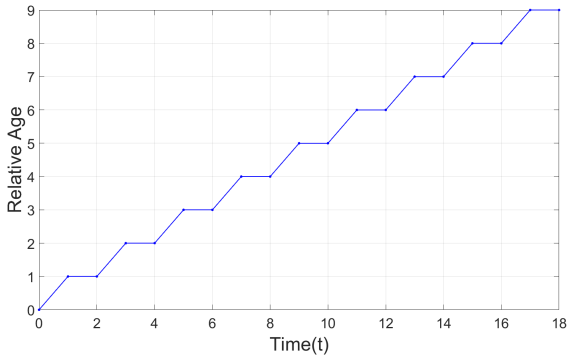
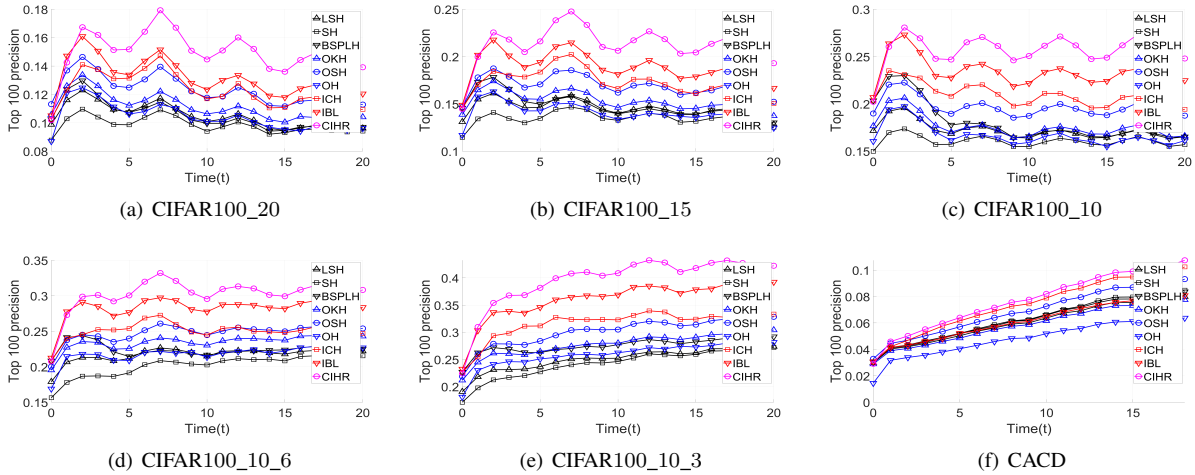


Fig. 9: Relative ages of face images in each data chunk over time.

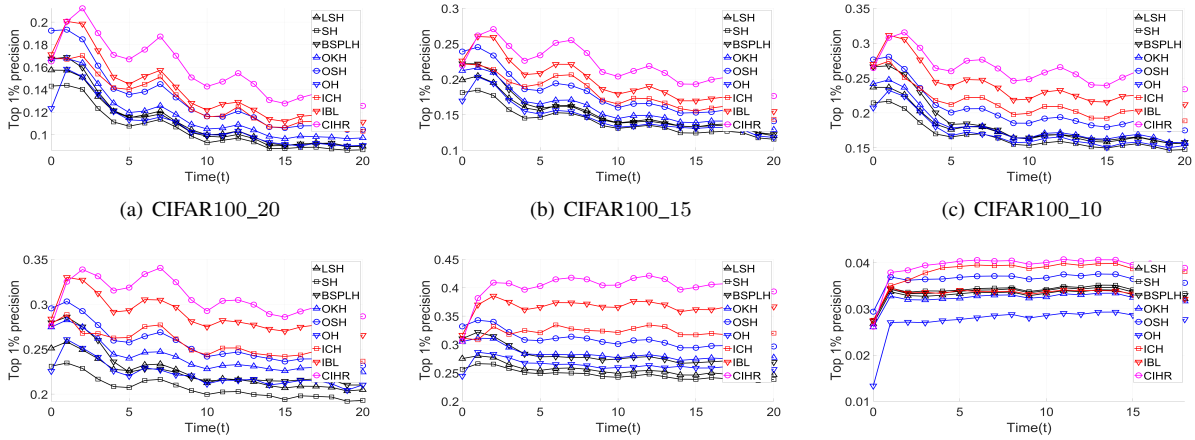
The face images appearing consecutively with growing age can also be viewed as the data distribution case. The Cross-Age Celebrity Dataset (CACD) is utilized in our experiments to simulate the situation data distribution drifting. This database contains $163,446$ images of $2,000$ celebrities. For each celebrity, face images are taken in consecutive ten years. Several samples are shown in Figure 8. For convenience, a feature called *relative age* is introduced for this database. The



(a) CIFAR100_20 (b) CIFAR100_15 (c) CIFAR100_10

(d) CIFAR100_10_6 (e) CIFAR100_10_3 (f) CACD

Fig. 10: Top 100 precision on 6 scenarios for data distribution drifting.



(a) CIFAR100_20 (b) CIFAR100_15 (c) CIFAR100_10

(d) CIFAR100_10_6 (e) CIFAR100_10_3 (f) CACD

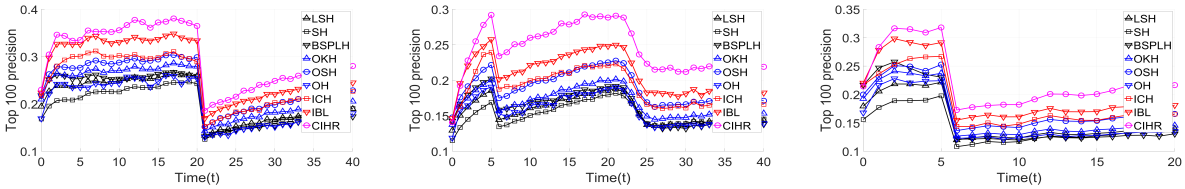
Fig. 11: Top 1% precision on 6 scenarios for data distribution drifting.

relative age of face images taken in the first year equals 0. The relative age of images taken in the second year is 1, and so on. In this way, the relative age of all face images in the database varies from 0 to 9. All images with relative age 0 are used to form the first data chunk at time $t = 0$, to make sure that there is no new celebrity appear afterwards. Two data chunks are generated for each relative age value. Thus, there are 18 chunks generated after time $t = 0$. Each data chunk consists of $4,000$ images, in which $2,000$ images are labeled. Moreover, a test set containing $1,000$ images is also generated at each time unit with age changing in the same manner. The relative ages of face images in each data chunk over time are shown in Figure 9. The settings of 6 non-stationary scenarios for data distribution drifting are summarized in Table III.

The experimental results of the hashing methods on 6 non-stationary scenarios for data distribution drifting are shown in Figure 10 and 11. For the five scenarios based on the CIFAR-100 database, the Top 1% precision of all hashing methods degrades over time due to the data distribution drifting problem. However, the degrees of degrade on the CIFAR100_10_6 and CIFAR100_10_3 scenarios, are visibly lighter than those on the CIFAR100_20, CIFAR100_15 and CIFAR100_10 scenarios. The reason is that fewer superclasses drift in CIFAR100_10_3 and CIFAR100_10_6 scenarios, comparing to the first three scenarios. Moreover, all hashing methods achieve local optimal performance at time $t = 3$, 8, 13, and 18, when one new subclass has higher appearance ratio than the previous major subclass. In all five scenarios, CIHR achieves best performance comparing to other hashing methods. For the CACD scenario, both top 100 precision and top 1% precision have a significant raise at time $t = 1$. That is because the number of each celebrity at time $t = 0$ is much smaller than 100 and 1% of the total amount of all images stored. Thus with relevant images increase over time, more relevant images are returned with smallest Hamming distances. The retrieval performances of online hashing methods OKH and OH are not satisfying. The reason is that these two methods update hash functions according to each new appearing data pairs with their similarity. For example, with a data chunk contacting 100 labeled images, only 50 pairwise similarities are utilized for updating hash functions, while the CIHR could use C_{100}^2 pairwise similarities for training. Thus OKH and

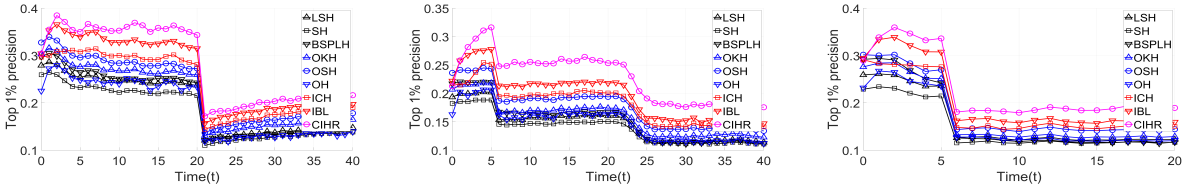
TABLE IV: Settings of 3 simulated scenarios for combined situations

Database	$0 \leq t \leq 20$	$21 \leq t \leq 40$	Scenario name
CIFAR-100	• 5 stationary superclasses • 5 superclasses drift when $0 \leq t \leq 20$	• 10 new superclasses appear	CIFAR100_DN
	• 15 stationary superclasses • 5 new superclasses appear since $t = 6$	• 10 out of 15 superclasses start drift • 5 new super-classes continuously appear	CIFAR100_ND
	• 5 stationary superclasses • 5 superclasses drift at $t = 0$ • 10 new superclasses appear since $t = 6$	• Experiment ends at $t = 20$	CIFAR100_D&N



(a) CIFAR100_DN (b) CIFAR100_ND (c) CIFAR100_D&N

Fig. 12: Top 100 precision on 3 scenarios for combined scenarios.



(a) CIFAR100_DN (b) CIFAR100_ND (c) CIFAR100_D&N

Fig. 13: Top 1% precision on 3 scenarios for combined scenarios.

OH are not suitable for the situation that new chunk of data appearing in batch manner. In CACD scenario, the proposed CIHR method achieves best retrieval performance.

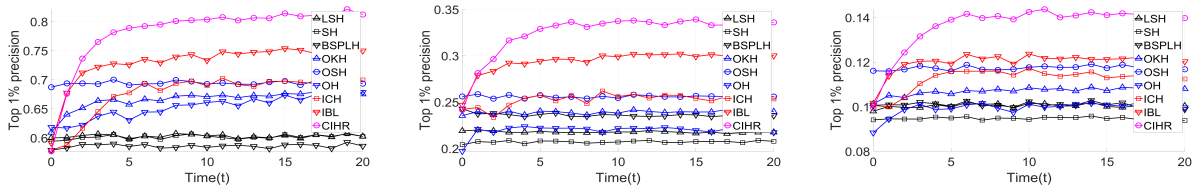
D. Combined Non-stationary Scenarios

Three non-stationary scenarios are simulated in our experiments for combined concept drift scenarios, i.e. new class appearing and data distribution drifting happening in order or simultaneously. For three scenarios, at each time unit, a data chunk with $1,000$ images is generated, which contains 100 labeled images. The settings of these three scenarios are shown in Table IV. A test set with $1,000$ images is also generated at each time unit following the same manner in Table IV. In the CIFAR100_DN scenario, 5 superclasses are used without data distribution drifting. Images of the one subclass in each of these superclasses are utilized to form the data chunks. 5 other randomly selected superclasses drift when $0 \leq t \leq 20$. After $t = 20$, images of 10 new superclasses start appears. These images are all selected from one subclass of each superclass. In the CIFAR100_ND scenario, 15 superclasses are used without data distribution drifting. Images of one subclass in each superclass are randomly selected when $0 <= t <= 20$. 5 new superclasses appear since time $t = 6$. Images in one subclass of each new superclass are used to build the new appearing data chunk. After time $t = 20$, images from 5 previously new appearing superclasses appear consecutively. Meanwhile, 10 out of 15 original superclasses starts to drift. In the CIFAR100_D&N scenario, two concept drift scenarios happen simultaneously. Five superclasses are stationary. Five other superclasses drift since time $t = 0$. Moreover, images from 10 new superclasses appear since $t = 6$.

The experimental results on 3 scenarios are shown in Figure 12 and 13. In the scenario CIFAR100_DN, there is a significant performance drop at time $t = 21$. The reason is that images from new classes appear at this time unit. The similar performance drop also happens at time $t = 6$ in the scenario CIFAR100_ND and CIFAR100_D&N, respectively. There is no significant drop due to the data distribution drifting in Figure 12(b) and 12(c). The reason may be that the degree of distribution drifting is slight. In all 3 scenarios, CIHR achieves the best retrieval accuracy over time.

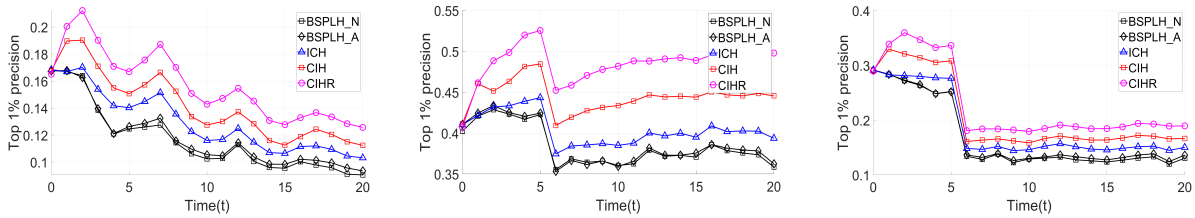
E. Non-stationary Scenarios without Concept Drift

The proposed method CIHR and comparative methods are also evaluated on three non-stationary scenarios without concept drift happening. Three databases, i.e. MNIST, CIFAR10, and CIFAR100 are utilized to simulate these three scenarios, i.e. MNIST_R, CIFAR10_R, and CIFAR100_R, respectively. For all three scenarios, a data chunk and a test set are generated at each time unit from time $t = 0$ to 20. Each data chunk



(a) MNIST_R (b) CIFAR10_R (c) CIFAR100_R

Fig. 14: Top 1% precision on 3 scenarios without concept drift.



(a) CIFAR100_20 (b) CIFAR10_1 (c) CIFAR100_D&N

Fig. 15: Top 1% Precision of CIHR and modified hashing methods.

contains 1000 images which are randomly selected from the original database. Test set at each time unit is generated in the same manner. Moreover, 100 images in each data chunk are randomly selected as labeled for training. Experimental results on 3 scenarios without concept drift are shown in Figure 14.

According to Figure 14, the performances of all hashing methods are relatively stable comparing to their performances on concept drift scenarios. Without new class appearing or data distribution drifting, each hashing method could learn comprehensive information for each class since time $t = 0$. Therefore, there is no retrieval performance drop happening afterwards. In all 3 scenarios, stationary hashing methods, i.e. LSH, SH, and BSPLH, train hash table at the beginning without updating and get worst retrieval performance. Among all non-stationary hashing methods, CIHR achieves the best retrieval performance in non-stationary data environment even without concept drift.

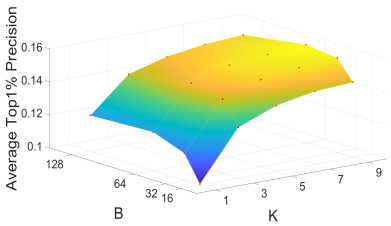
F. Comparison with Modified Hashing methods

To further evaluate the effectiveness of the proposed method, CIHR is also compared with two modified versions of an existing stationary hashing method, i.e. BSPLH, which is also used to train individual hash table in CIHR. Based on BSPLH, BSPLH_N retrains its hash table based on the newest data chunk only when new chunk of data appears, while BSPLH_A retrains its hash table using all existing data stored in database at each time unit. For both methods, only the labeled images in the newest data chunk, which contain current data distribution information of each class, are utilized as supervised information for training. Moreover, ablation experiment comparison is also shown in this section, to validate the influences of different parts in CIHR (i.e. training complementary hash tables and query-adaptive re-ranking) to the final retrieval performance. Comparing to CIHR which trains complementary hash tables with using query-adaptive re-ranking, ICH trains new hash table independently without using query-adaptive re-ranking. Thus ICH is used as baseline method. A simplified version of CIHR, i.e. CIH, is also compared, which only trains complementary hash tables without using query-adaptive re-ranking. All these hashing methods are tested on 3 representative scenarios, i.e. CIFAR100_20, CIFAR10_1, and CIFAR100_D&N.

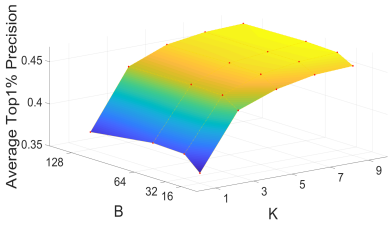
The Top 1% precision of all hashing methods are shown in Figure 15. BSPLH_N and BSPLH_A update hash tables over time in different ways. BSPLH_N ignores the data information in the previous data chunks. BSPLH_A updates hash table based on all existing data, in which the newest data only accounts for a small portion. Therefore, both methods cannot adapt to the non-stationary data environment well. ICH trains new hash table and updates multi-hashing system at each time unit. Thus ICH achieves better retrieval performance than methods of simply retraining hash table using new data, i.e. BSPLH_N and BSPLH_A. However, ICH trains new hash table independently without considering the performance of previous hash tables. CIH achieves better performance than the original ICH method, which validate the effectiveness of training complementary hash tables. With query-adaptive re-ranking used, the proposed CIHR method gets a significant improvement of retrieval performance than CIH and achieves the best performance in all 3 scenarios. This proves the effectiveness of the query-adaptive weights in CIHR.

G. Parameter Selection

In the proposed CIHR method, there are five parameters concerned, i.e. number of hash tables (K), length of hash code (B), threshold parameter for similar images (α_s), threshold parameter for dissimilar images (α_d), and number of stored badly hashed images (n). The CIHR method with different values of parameters are evaluated on 2 non-stationary scenarios CIFAR100_20 and CIFAR10_1. Figure 16 shows the

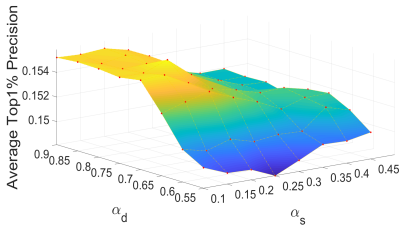


(a) CIFAR100_20

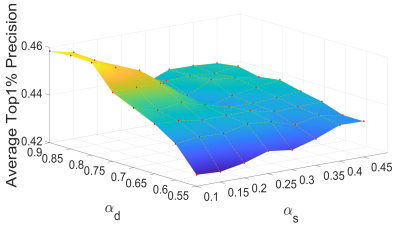


(b) CIFAR10_1

Fig. 16: Average Top 1% precision of the CIHR with different values of K and B.



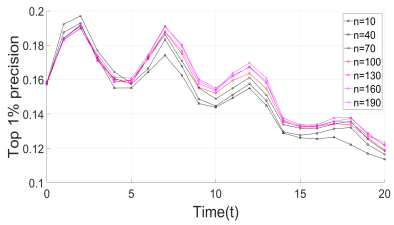
(a) CIFAR100_20



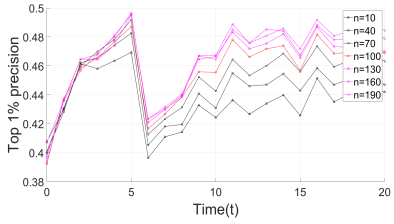
(b) CIFAR10_1

Fig. 17: Average Top 1% precision of the CIHR with different values of α_s and α_d.

average Top 1% precision of CIHR from time $t = 0$ to 20 with different values of K and B. Considering both retrieval performance and space costs, we set $K = 5$ and $B = 64$ in all our experiments. In Figure 17, the average Top 1% precision of CIHR during time $t = 0$ to 20 is shown with different values of α_s and α_d. These two parameters control two threshold Hamming distances for determining badly hashed images. According to Figure 17, CIHR with large threshold Hamming distance for dissimilar images and small threshold Hamming distance for similar images achieves nearly optimal retrieval performance. Thus, we set $\alpha_s = 0.1$, $\alpha_d = 0.9$ in all our experiments. Moreover, we also notice that CIHR gets nearly the worst performance when these two parameters are both small. The reason may be that more image pairs are wrongly evaluated as dissimilar, which causes more errors for images in database and make it more difficult to select real badly hashed images. The Top 1% precision of CIHR with different values of n is shown in Figure 18. We set $n = 100$ in all our experiments, since the performance increasement is not significant when n gets bigger.



(a) CIFAR100_20



(b) CIFAR10_1

Fig. 18: Top 1% precision of the CIHR with different values of n.

V. Conclusions

In this paper, a novel image hashing method, CIHR, is proposed for large scale image retrieval in non-stationary environments with concept drifts. It builds a multi-hashing system, which consists of multiple hash tables that are created incrementally as images become available in successive chunks. A new chunk of images, combined with images in previous chunks that are badly hashed by previous hash tables, are used as the training data set to learn a new hash table. Furthermore, a query-adaptive re-ranking method is also proposed to evaluate each hash function according to the location of the query with respect to the corresponding hash hyperplane. The proposed hashing method CIHR is evaluated on simulated non-stationary scenarios, and achieves superior retrieval performance over the other state-of-the-art non-stationary hashing methods.

In CIHR, a new hash table is trained whenever a new chunk of data appears. However, concept drift may not happen with every chunk of data. Therefore, a future work is to add the detection function of concept drift into CIHR, so that a new hash table is trained only when a concept drift is detected. Recent years have also witnessed the rapid development of active learning. Considering the newly appearing data are mostly unlabeled and maybe redundant in non-stationary data environment, applying active learning strategy in CIHR for training new hash table will be an interesting future topic. Moreover, transfer learning transfers knowledge of datasets with different distributions. Applying transfer learning tech-

nique to train new hash table at each time unit is another valuable idea. In this way, the information learnt based on old data would also be helpful for the training of new hash tables. Further researches on the fusion of these techniques to CIHR are needed. Last but not the least, existing non-stationary hashing methods are mostly based on hand-crafted features, and current deep hashing methods are still focused on the retrieval task in static data environments. An interesting open research question is how to update hash tables using deep neural networks to adapt to non-stationary data environments with concept drifts.

REFERENCES

[1] X. Liu, G. Cheung, C.-W. Lin, D. Zhao, and W. Gao, "Prior-based quantization bin matching for cloud storage of jpeg images," *IEEE Transactions on Image Processing*, vol. 27, no. 7, pp. 3222–3235, 2018.

[2] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Trans. Neural Networks*, vol. 22, no. 10, pp. 1517–1531, 2011.

[3] W. W. Ng, X. Tian, Y. Lv, D. S. Yeung, and W. Pedrycz, "Incremental hashing for semantic image retrieval in nonstationary environments," *IEEE Trans. Cybern.*, vol. 47, no. 11, pp. 3814–3826, 2017.

[4] L.-K. Huang, Q. Yang, and W.-S. Zheng, "Online hashing," in *Proc. of International Joint Conference on Artificial Intelligence*, 2013, pp. 1422–1428.

[5] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality sensitive hashing scheme based on p-stable distributions," in *Proc. of the Twentieth Annual Symposium on Computational Geometry*, 2004, pp. 253–262.

[6] M. Raginsky and S. Lazebnik, "Locality-sensitive binary codes from shift-invariant kernels," in *Proc. of Conf. and Workshop on Neural Information Processing Systems*, 2009, pp. 1509–1517.

[7] K. Jiang, Q. Que, and B. Kulis, "Revisiting kernelized locality-sensitive hashing for improved large-scale image retrieval," in *Proc. IEEE Conf. Comput. Vis. Patt. Recognit. IEEE*, 2015, pp. 4933–4941.

[8] D. Gorisse, M. Cord, and F. Precioso, "Locality-sensitive hashing for chi2 distance," *IEEE Trans. on Pattern Anal. Mach. Intell.*, vol. 34, no. 2, pp. 402–409, 2012.

[9] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. of Advances in Neural Information Processing Systems*, 2009, pp. 1753–1760.

[10] D. Hu, F. Nie, and X. Li, "Discrete spectral hashing for efficient similarity retrieval," *IEEE Transactions on Image Processing*, vol. 28, no. 3, pp. 1080–1091, 2019.

[11] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon, "Spherical hashing: binary code embedding with hyperspheres," *IEEE Trans. on Pattern Anal. Mach. Intell.*, vol. 37, no. 11, pp. 2304–2316, 2015.

[12] L. Zhu, J. Shen, L. Xie, and Z. Cheng, "Unsupervised visual hashing with semantic assistant for content-based image retrieval," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 2, pp. 472–486, 2017.

[13] L. Liu, M. Yu, and L. Shao, "Unsupervised local feature hashing for image similarity search," *IEEE Trans. Cybern.*, pp. 1–11, 2015.

[14] R. Ye and X. Li, "Compact structure hashing via sparse and similarity preserving embedding," *IEEE Trans. Cybern.*, vol. 46, no. 3, pp. 718–729, 2016.

[15] Y. Guo, G. Ding, L. Liu, J. Han, and L. Shao, "Learning to hash with optimized anchor embedding for scalable retrieval," *IEEE Trans. on Image Processing*, vol. 26, no. 3, pp. 1344–1354, 2017.

[16] S. Wang, C. Li, and H.-L. Shen, "Distributed graph hashing," *IEEE Transactions on Cybernetics*, vol. 49, no. 5, pp. 1896–1908, 2019.

[17] W. Liu, J. Wang, S. Kumar, and S.-F. Chang, "Hashing with graphs." IEEE, 2011, pp. 1–8.

[18] W. Kong, W.-J. Li, and M. Guo, "Manhattan hashing for large-scale image retrieval." IEEE, 2012, pp. 45–54.

[19] M. Wang, H. Li, D. Tao, K. Lu, and X. Wu, "Multimodal graph-based reranking for web image search," *IEEE Transactions on Image Processing*, vol. 21, no. 11, pp. 4649–4661, 2012.

[20] M. Wang, W. Li, D. Liu, B. Ni, J. Shen, and S. Yan, "Facilitating image search with a scalable and compact semantic mapping," *IEEE Transactions on Cybernetics*, vol. 45, no. 8, pp. 1561–1574, 2015.

[21] X. Liu, L. Huang, C. Deng, B. Lang, and D. Tao, "Query-adaptive hash code ranking for large-scale multi-view visual search," *IEEE Trans. Image Processing*, vol. 25, no. 10, pp. 4514–4524, 2016.

[22] X. Liu, J. He, B. Lang, and S.-F. Chang, "Hash bit selection: a unified solution for selection problems in hashing," in *Proc. IEEE Conf. Comput. Vis. Patt. Recognit.*, 2013, pp. 1570–1577.

[23] S. Zhang, M. Yang, T. Cour, K. Yu, and D. N. Metaxas, "Query specific rank fusion for image retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 4, pp. 803–815, 2014.

[24] X. Liu, C. Deng, B. Lang, D. Tao, and X. Li, "Query-adaptive reciprocal hash tables for nearest neighbor search," *IEEE Transactions on Image Processing*, vol. 25, no. 2, pp. 907–919, 2015.

[25] P. Li, M. Wang, J. Cheng, C. Xu, and H. Lu, "Spectral hashing with semantically consistent graph for image indexing," *IEEE Trans. Multimedia*, vol. 15, no. 1, pp. 141–152, 2013.

[26] Z. Chen, J. Lu, J. Feng, and J. Zhou, "Nonlinear discrete hashing," *IEEE Trans. Multimedia*, vol. 19, no. 1, pp. 123–135, 2017.

[27] D. Wang, X. Gao, X. Wang, L. He, and B. Yuan, "Multimodal discriminative binary embedding for large-scale cross-modal retrieval," *IEEE Trans. Image Processing*, vol. 25, no. 10, pp. 4540–4554, 2016.

[28] J. Tang, K. Wang, and L. Shao, "Supervised matrix factorization hashing for cross-modal retrieval," *IEEE Trans. Image Processing*, vol. 25, no. 7, pp. 3157–3166, 2016.

[29] Y. Zhen, Y. Gao, D.-Y. Yeung, H. Zha, and X. Li, "Spectral multimodal hashing and its application to multimedia retrieval," *IEEE Trans. cybernetics*, vol. 46, no. 1, pp. 27–38, 2016.

[30] H. Liu, R. Wang, S. Shan, and X. Chen, "Deep supervised hashing for fast image retrieval," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2064–2072.

[31] H. Lai, P. Yan, X. Shu, Y. Wei, and S. Yan, "Instance-aware hashing for multi-label image retrieval," *IEEE Transactions on Image Processing*, vol. 25, no. 6, pp. 2469–2479, 2016.

[32] C. Deng, Z. Chen, X. Liu, X. Gao, and D. Tao, "Triplet-based deep hashing network for cross-modal retrieval," *IEEE Transactions on Image Processing*, vol. 27, no. 8, pp. 3893–3903, 2018.

[33] J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for large scale search," *IEEE Trans. on Pattern Anal. Mach. Intell.*, vol. 34, no. 12, pp. 2393–2406, 2012.

[34] C. Wu, J. Zhu, D. Cai, C. Chen, and J. Bu, "Semi-supervised nonlinear hashing using bootstrap sequential projection learning," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 6, pp. 1380–1393, 2013.

[35] H. Xu, J. Wang, Z. Li, G. Zeng, S. Li, and N. Yu, "Complementary hashing for approximate nearest neighbor search," in *Proc. of IEEE Int. Conf. on Computer Vision*. IEEE, 2011, pp. 1631–1638.

[36] P. Li, J. Cheng, and H. Lu, "Hashing with dual complementary projection learning for fast image retrieval," *Neurocomputing*, vol. 120, pp. 83–89, 2013.

[37] W. W. Ng, X. Zhou, X. Tian, X. Wang, and D. S. Yeung, "Bagging-boosting-based semi-supervised multi-hashing with query-adaptive re-ranking," *Neurocomputing*, vol. 275, no. 31, pp. 916–923, 2018.

[38] F. Cakir and S. Sclaroff, "Online supervised hashing," in *Image Processing (ICIP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 2606–2610.

[39] ——, "Adaptive hashing for fast similarity search," in *Proc. of IEEE Int. Conf. on Computer Vision*, 2015, pp. 1044–1052.

[40] F. Cakir, K. He, S. A. Bargal, and S. Sclaroff, "Mihash: Online hashing with mutual information," in *Proc. of IEEE Int. Conf. on Computer Vision*, 2017, pp. 437–445.

[41] C. Leng, J. Wu, J. Cheng, X. Bai, and H. Lu, "Online sketching hashing," in *Proc. IEEE Conf. Comput. Vis. Patt. Recognit.*, 2015, pp. 2503–2511.

[42] Z. Weng and Y. Zhu, "Online supervised sketching hashing for large-scale image retrieval," *IEEE Access*, vol. 7, pp. 88 369–88 379, 2019.

[43] L. Huang, Q. Yang, and W. S. Zheng, "Online hashing," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 6, pp. 2309–2322, 2018.

[44] W. W. Ng, X. Tian, W. Pedrycz, X. Wang, and D. S. Yeung, "Incremental hash-bit learning for semantic image retrieval in nonstationary environments," *IEEE transactions on cybernetics, early access*, 2018.

[45] B. C. Chen, C. S. Chen, and W. H. Hsu, "Face recognition and retrieval using cross-age reference coding with cross-age celebrity dataset," *IEEE Transactions on Multimedia*, vol. 17, no. 6, pp. 804–815, 2015.

[46] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, 2016.

[47] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proc. IEEE Conf. Comput. Vis. Patt. Recognit.*, 2015, pp. 815–823.